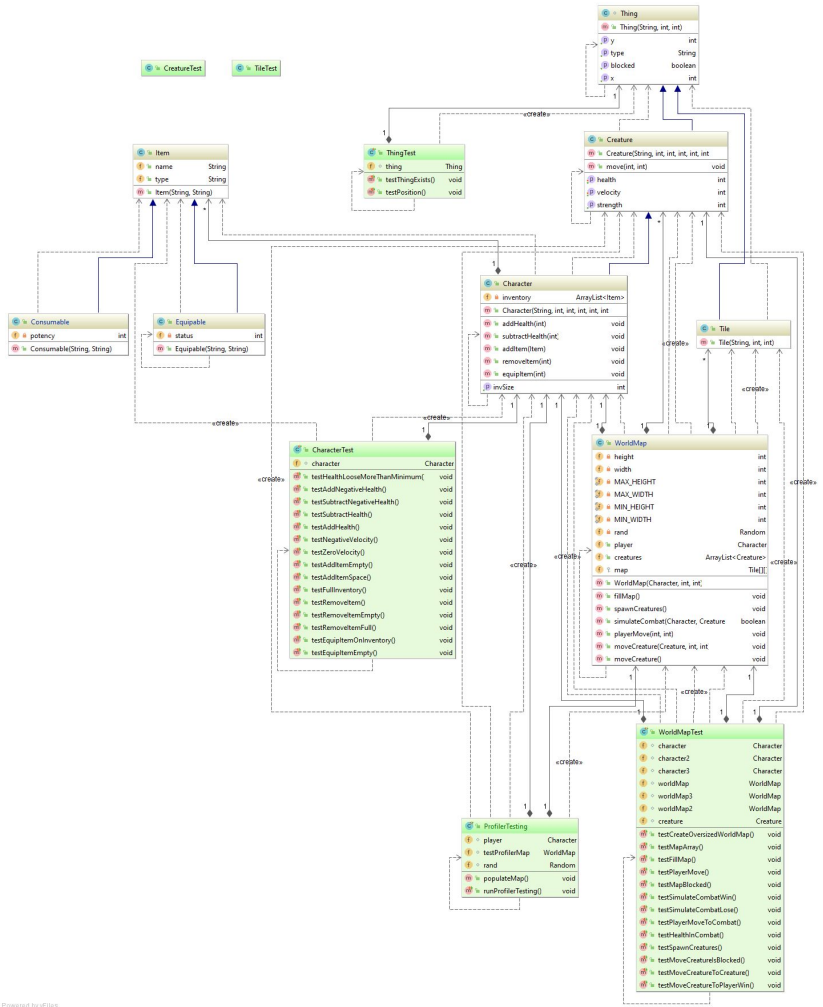
Grupp nr: 1

- Leo Blumenberg, leo.blumenberg@gmail.com
- Markus Bowie, markus.bowie@gmail.com
- Noël Hennings, noel.hennings@gmail.com

Verktyg

- Git
- IntelliJ
- Eclipse
- JUnit
- JProfiler
- FindBugs
- SpotBugs
- Maven
- MetricsReloaded

Slutlig design



TDD-exempel: Subtract Health

Testkod

```
@Test
public void testSubtractHealth() {
   character.subtractHealth(50);
   assertEquals(50, character.getHealth());
}
```

Koden som testas

```
public void subtractHealth(int
 healthToBeSubtracted) {
   health = health - healthToBeSubtracted;
}
```

Markus

TDD-exempel: Negative health

Testkod

```
@Test
public void
 testHealthLooseMoreThanMinimum() {

 character.subtractHealth(character
 .getHealth() + 1);
    assertEquals(-1,
 character.getHealth());
}
```

Koden som testas

```
public void subtractHealth(int
 healthToBeSubtracted) {
   health = health - healthToBeSubtracted;
}
```

Markus

TDD-exempel: Negative velocity

Testkod

```
@Test
@DisplayName("Test Negative Velocity")
public void testNegativeVelocity() {
   character.setVelocity(character.getVelocity() -
 101);
   assertEquals(-1, character.getVelocity());
}
```

Koden som testas

```
public void setVelocity(int velocity) {
   this.velocity = velocity;
}
```

Noel

TDD-exempel: Move creature while tile is blocked

Testkod

```
@Test
@DisplayName("Move creature while tile is blocked")
public void testMoveCreatureIsBlocked(){
   worldMap.spawnCreatures();
   worldMap.moveCreature(worldMap.creatures.get(0),
-1,0);
   assertEquals(1,
worldMap.creatures.get(0).getX());
   assertEquals(500,
worldMap.creatures.get(0).getY());
}
```

Koden som testas

```
public void moveCreature(Creature creature, int x,
int y){
   boolean creatureBlock = false;
   int oldX = creature.getX();
   int oldY = creature.getY();
   Tile tile = map[oldX + x][oldY + y];
   for(Creature iter : creatures){
       if(iter.getX() == (oldX + x) && iter.getY()
== (oldY + y)){
           creatureBlock = true;
       }
   }

   if (!tile.isBlocked() && !creatureBlock)
       creature.move(x,y);

if(player.getX()==creature.getX()&&player.getY()==cr
eature.getY())
           if(simulateCombat(player, creature))
               creatures.remove(creature);
}
```

Noel

TDD-exempel: Test remove item from inventory

Testkod

```
@Test
@DisplayName("Test remove item from
Inventory")
public void testRemoveItem(){
   character.addItem(new
Item("Sword","Weapon"));
   character.removeItem(0);
   assertEquals(0,
character.getInvSize());
```

Koden som testas

```
public void removeItem(int i){
   if(inventory.size()==0)
       throw new
IllegalArgumentException("Nothing
to remove");
   else
       inventory.remove(i);
}
```

Leo

TDD-exempel: Move creature to creature

Leo

Testkod

```
@Test
@DisplayName("Move creature to creature")
public void testMoveCreatureToCreature(){
   worldMap.spawnCreatures();

worldMap.moveCreature(worldMap.creatures.get
(1),-1,0);

worldMap.moveCreature(worldMap.creatures.get
(2),0,0);
   assertEquals(500,
worldMap.creatures.get(1).getX());
   assertEquals(500,
worldMap.creatures.get(1).getY());
   assertEquals(499,
worldMap.creatures.get(2).getX());
   assertEquals(500,
worldMap.creatures.get(2).getY());
}
```

Koden som testas

```
public void moveCreature(Creature creature,
int x, int y){
   boolean creatureBlock = false;
   int oldX = creature.getX();
   int oldY = creature.getY();
   Tile tile = map[oldX + x][oldY + y];
   for(Creature iter : creatures){
       if(iter.getX() == (oldX + x) &&
iter.getY() == (oldY + y)){
           creatureBlock = true;
       }
   }

   if (!tile.isBlocked() && !creatureBlock)
       creature.move(x,y);

if(player.getX()==creature.getX()&&player.ge
tY()==creature.getY())
           if(simulateCombat(player,
creature))
               creatures.remove(creature);
}
```

TDD erfarenheter

I början byggde vi tester och sedan kod utan att först ha använt oss av testdesigntekniker. När vi väl gjorde det insåg vi hur mycket det underlättade. Det blev lättare att bygga både testerna och klassen.

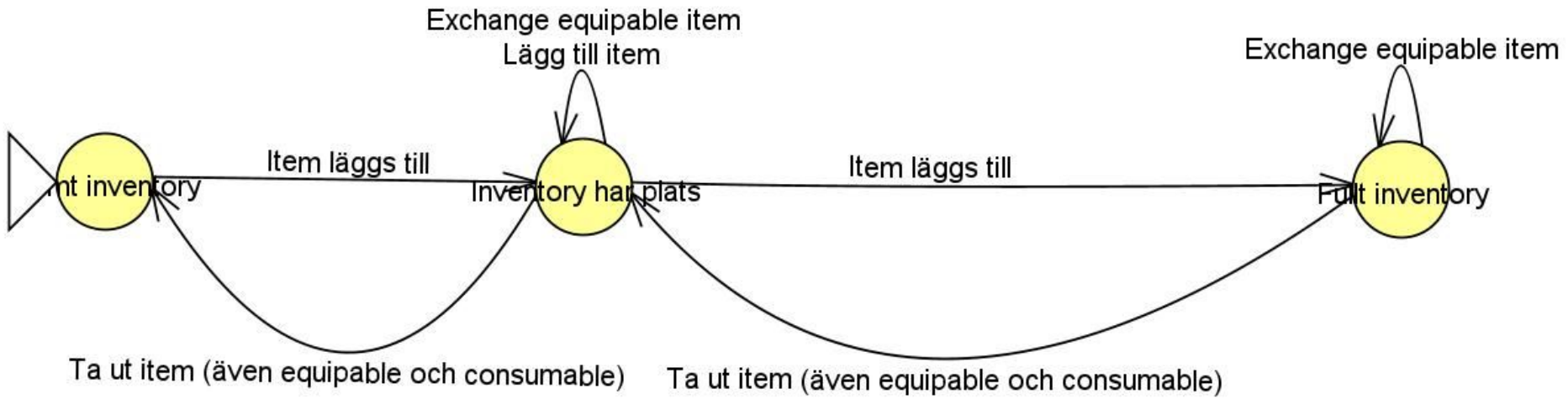
Det tog ett tag för oss att ställa om till att jobba testdrivet eftersom vi var vana vid att bygga själva koden först.

Tillståndsmaskiner

Till en början försökte vi göra en beslutstabell på Inventory men det var svårt att hitta bra villkor på den och få fram testfall. Till slut tillämpade vi istället en tillståndsmaskin. Det var lämpligt eftersom Inventory har tydliga tillstånd: Tomt, Fullt och Plats finns.

Tillståndsmaskinen

Tillståndsmaskin över inventory



Testfall

Vi gjorde testfall till addItem och till removeItem. Vi började jobba med equipable items men eftersom vi fick ont om tid valde vi inte utveckla dem mer.

Klass	Testfall
CharacterTest	Test Add item to empty Inventory
CharacterTest	Test Add Item to an Inventory with space
CharacterTest	Test adding more Items than Inventory slots
CharacterTest	Test remove item from Inventory
CharacterTest	Test Remove Item from empty Inventory
CharacterTest	Test Remove Item from Full Inventory

Beslutstabeller/beslutsträd

Från den formella granskningen kom vi fram till att det var oklart hur MoveCreature skulle fungera och att det eventuellt kunde uppstå overflow. T. ex. vad som händer när fler creatures befinner sig på samma tile och vad som händer när spelare försöker slåss mot flera creatures samtidigt. Denna beslutstabell hjälpte oss komma till klarhet med detta.

Beslutstabell

	R1	R2	R3	R4	R5	R6	R7	R8
Tile is blocked	T	T	T	T	F	F	F	F
Tile has creature	T	T	F	F	T	T	F	F
Tile has player	T	F	T	F	T	F	T	F
Resultat								
Move Action	-	-	-	-	-	-	y	y
Combat	-	-	-	-	-	-	y	n

	R1	R2	R3	R4
Tile is blocked	F	F	F	F
Tile has creature	T	T	F	F
Tile has player	T	F	T	F
Resultat				
Move Action	-	-	y	y
Combat	-	-	y	n

Testfall

Vi gjorde testfall för moveCreature i WorldMap klassen.

Klass	Testfall
WorldMap	Move creature while tile is blocked
WorldMap	Move creature to creature
WorldMap	Move creature to player, start battle and player wins

Granskning

Vi valde att genomföra granskningen på hela koden eftersom det var liten. Vi baserade vår granskning på checklistan som vi blev tilldelade på seminarie 2.

Granskning

Class Definition

1. Does each class have a descriptive name used in accord with naming conventions?

   Ja

2. For each class and interface: Are the declarations ordered as follows: Class/interface documentation comment, class or interface statement, Class/interface implementation comment, Class (static) variables and constants, Instance variables, Constructors, Methods?

   I WorldMap.java ligger de statiska variablerna blandade med instansvariablerna. Metodernas ordning är förvirrande. I övrigt korrekt.

3. Does each class have an appropriate constructor?

   Alla utom Consumable.java

4. For each member of every class: Could access to the member be further restricted?

   Ja: Mest i Thing-trädet. Character.java kan vara package private. Även Creature.java.

   Creature kan eventuellt vara abstract. Eventuellt Item abstract. Thing bör vara abstract.

5. Do any derived classes have common members that should be in the base class?

   Nej.

6. Can the class inheritance hierarchy be simplified?

   Nja. Möjligtvis att Equipable och Consumable inte behöver vara separata klasser.

Method

7. Are descriptive method names used in accord with naming conventions?

   Ja.

8. Is every method parameter value checked before being used?

   Nej, saknas totalt.

9. For every method: Does it return the correct value at every method return point?

   Antagligen. Bör kontrolleras.

10. Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?

    Ja. Variabeln Blocked i Thing.java borde inte vara i konstruktorn. I Item.java finns en String type-dublett från Thing.java.

11. Do the values in units agree (e.g., euro vs. cents)?

    N/A.

12. Are there static methods that should be non-static or vice-versa?

    Nej.

Declarations

13. Are descriptive variable and constant names used in accord with naming conventions?

    Ja.

14. Are there variables with confusingly similar names?

    Nej.

15. Is every variable and attribute correctly typed?

    Ja.

16. Is every variable and attribute properly initialized?

    Variabeln Blocked i Thing.java borde inte vara i konstruktorn.

17. Could any non-local variables be made local?

```
I WorldMap.java: public Creature player;
public Set<Creature> creatures;
protected Tile[][] map;
kan vara private eller package private.
```

18. Are there literal constants that should be named constants?

    No. Eventuellt inventory, om man inte vill kunna ändra den.

19. Are there variables that should be constants?

    No.

Computation & Comparison

20. Is overflow or underflow possible during a computation?

    Möjligen i simulate combat-metoden, om player och två creatures hamnar på samma tile.

21. For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?

    Ja.

22. Are parentheses used to avoid ambiguity?

    Nej.

23. Are the comparison operators correct?

    N/A

24. Is each boolean expression correct?

    Ja.

25. Are there improper and unnoticed side-effects of a comparison?

    N/A

Control Flow

26. For each loop: Is the best choice of looping constructs used?

    Ja.

27. Will all loops terminate?

    Nej, simulate combat kan fastna om båda har noll i strength.

28. When there are multiple exits from a loop, is each exit necessary and handled properly?

    Ja.

29. Does each switch statement have a default case?

    Nej, WorldMap.java --> fillMap() saknar ett.

30. Are missing switch case break statements correct and marked with a comment?

    N/A. Alla cases har break statements.

31. Is the nesting of loops and branches too deep, and is it correct?

    Korrekt.

32. Can any nested if statements be converted into a switch statement?

    Nej. Redan gjort med fillMap.

33. Are null bodied control structures correct and marked with braces or comments?

    N/A

34. Does every method terminate?

    Ja.

Performance

35. Can better data structures or more efficient algorithms be used?

    Möjligen. Vi har använt arrayer. Kanske HashMaps, Listor eller Sets hade varit mer optimalt. simulateCombat är lite diffus.

36. Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?

    Detta har vi ej gjort.

37. Can the cost of recomputing a value be reduced by computing it once and storing the results?

    Nej, görs redan med t. ex. health.

38. Is every result that is computed and stored actually used?

    Ja.

39. Can a computation be moved outside a loop?

    Nej.

40. Are there tests within a loop that do not need to be done?

    Nej.

41. Can a short loop be unrolled?

    Nej.

42. Are there two loops operating on the same data that can be combined into one?

    Nej.

Comments

43. Does every method and class have an appropriate header comment?

    Nej.

44. Does every variable and constant declaration have a comment?

    Nej.

45. Is the underlying behavior of each method and class expressed in plain language?

    Nej.

46. Is the header comment for each method and class consistent with the behavior of the method or class?

    N/A.

47. Is only specified functionality implemented with no additional functionality added?

    N/A.

48. Do the comments and code agree?

    N/A.

49. Do the comments help in understanding the code?

    N/A.

50. Are there enough comments in the code?

    No!

51. Are there too many comments in the code?

    N/A.

Packaging

52. For each file: Does it contain only one class?

    Ja.

53. For each file: Does it have the following ordering: Package and Import statements, beginning comments, Class and Interface declaration?

    Ja. Dock inga kommentarer.

54. Does each line contain at most one statement?

    Ja.

55. For each line: Is it no more than about 80 characters long?

    Nej, längre på sina håll. T. ex. addHealth i Character

56. For each method: Is it no more than about 60 lines long?

    Stämmer.

57. For each class: Is no more than 2000 lines long?

    N/A

Modularity

58. Is there a low level of coupling between packages and/or classes?

    Nej.

59. Is there a high level of cohesion within each package?

    Ja, men WorldMap gör lite för mycket.

60. Is there duplicate code that could be replaced by a call to a method that provides the behavior of the duplicate code?

    Position duplicate code.

61. Are framework classes used where and when appropriate?

    Ja.

Granskningsrapport

På de föregående sidorna har vi bifogat alla våra påträffade fel och här följer urval.

Class Definition

2 Följer inte konventionen men programmet fungerar ändå korrekt. (1)

3 Klassen Consumable saknar konstruktor. (5)

4 Åtkomst kan begränsas ytterligare. (3)

Metod

8 Inga null-kontroller. (5)

9 Borde kontrollera att metoderna returnerar korrekta värden. (3)

10 Borde städas upp. (2)

Declarations

16 Variabeln Blocked i Thing.java borde inte vara i konstruktorn. (3)

17 Variabler som borde vara local. (3)

Computation and Comparison

20 Combat kan få spelet att krascha. (5)

Skala

1 = Inte allvarligt

2 = Inte särskilt allvarligt

3 = Allvarligt

4 = Mycket allvarligt

5 = Kritiskt

Control Flow

27 Osannolikt att det sker men det är möjligt och isåfall allvarligt. (3)

29 Switchen fungerar även utan default case. (1)

Performance

35 Det är möjligt att man kan optimera här. (2)

Comments

43-45, 50 Vi har nästan inga kommentarer alls. (5)

Packaging

55 Kan kortas ner. (2)

Modularity

59 WorldMap gör för mycket. (4)

60 Position duplicate code. (4)

Skala

1 = Inte allvarligt

2 = Inte särskilt allvarligt

3 = Allvarligt

4 = Mycket allvarligt

5 = Kritiskt

Erfarenheter av granskning

Tack vare checklistan från seminariet hittade vi fel som vi inte ens visste att vi borde titta efter.

När vi gick igenom checklistan var det många saker som var N/A, vilket gjorde oss medvetna om att vi inte hade tillämpat TDD i tillräckligt stor utsträckning.

Den här typen av formaliserad granskning leder definitivt till att man hittar fler fel än vid granskning av mer informell karaktär. Kanske i synnerhet för oss nybörjare.

Kodkritiksystem

Handlar om indikationer. Sannolikhet. Smart och kostnadseffektivt att köra under arbetets gång, för att undvika att man bygger in buggar i systemet.

Kodkritiksystem

FindBugs fungerande eventuellt inte, eftersom den inte hittade några buggar vare sig i vår gamla eller nya version. Se följande slides. Teoretiskt sett kan det förstås vara så att vår kod är (och var) buggfri, men det verkar osannolikt.

För att försöka få FindBugs att ge utslag prövade vi att höja känslighetsnivån, men inget hände. Vi testade även olika plug-ins: FindBugs-IDEA och QA-plugs FindBugs för IntelliJ utan framgång. Vi testade även en standalone version med ett GUI som inte ville analysera filerna.

Kodkritiksystem

Slutligen testade vi SpotBugs för Eclipse vilket gav ett utslag i den nya versionen. Buggen låg i konstruktorn i klassen Equipable, som ändå inte var implementerat.

På den gamla versionen hittade SpotBugs för Eclipse samma bug.

Kodkritiksystem

SENASTE VERSIONEN, EFTER GRANSKNING

FindBugs-IDEA: found 0 bugs in 0 class
using FindBugs-IDEA 1.0.1 with Findbugs version 3.0.1

InteRouge[InteRouge]

Configured Output Files/Paths - the analysis entry point (14)
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Item.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Tile.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Thing.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Creature.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/WorldMap.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Character.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Equipable.java
C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Consumable.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/TileTest.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/ThingTest.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/CreatureTest.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/WorldMapTest.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/CharacterTest.java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/ProfilerTesting.java

Compile Output Path (IDEA)
C:/Users/Ingela/INTE_Project/Rouge/src/main/java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java

Sources Dir List (2)
C:/Users/Ingela/INTE_Project/Rouge/src/main/java
C:/Users/Ingela/INTE_Project/Rouge/src/test/java

Analyzed Classes List (14)
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Item.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Tile.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Thing.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Creature.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\WorldMap.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Character.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Equipable.class
C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Consumable.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\TileTest.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\ThingTest.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\CreatureTest.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\WorldMapTest.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\CharacterTest.class
C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\ProfilerTesting.class

Aux Classpath Entries (2)
C:/Users/Ingela/INTE_Project/Rouge/target/classes
C:/Users/Ingela/INTE_Project/Rouge/target/test-classes

Kodkritiksystem

GAMLA VERSIONEN, FÖRE GRANSKNING

FindBugs-IDEA: found 0 bugs in 0 class

using FindBugs-IDEA 1.0.1 with Findbugs version 3.0.1

InteRouge[InteRouge]

Configured Output Files/Paths - the analysis entry point (12)

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Item.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Tile.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Thing.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Creature.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Position.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/WorldMap.java

C:/Users/Ingela/INTE_Project/Rouge/src/main/java/rougelikeclasses/Character.java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/TileTest.java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/ThingTest.java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/CreatureTest.java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/WorldMapTest.java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java/rougelikeclasses/CharacterTest.java

Compile Output Path (IDEA)

C:/Users/Ingela/INTE_Project/Rouge/src/main/java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java

Sources Dir List (2)

C:/Users/Ingela/INTE_Project/Rouge/src/main/java

C:/Users/Ingela/INTE_Project/Rouge/src/test/java

Analyzed Classes List (12)

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Item.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Tile.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Thing.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Creature.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Position.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\WorldMap.class

C:\Users\Ingela\INTE_Project\Rouge\target\classes\rougelikeclasses\Character.class

C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\TileTest.class

C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\ThingTest.class

C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\CreatureTest.class

C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\WorldMapTest.class

C:\Users\Ingela\INTE_Project\Rouge\target\test-classes\rougelikeclasses\CharacterTest.class

Aux Classpath Entries (2)

C:/Users/Ingela/INTE_Project/Rouge/target/classes

C:/Users/Ingela/INTE_Project/Rouge/target/test-classes

Statiska mått

Gamla versionen

Method metrics

	essential cyclomatic complexity(G)	design complexity(G)	cyclomatic complexity(G)
rougelikeclasses.WorldMap.fillMap()	2	5	11

Den har en cyclomatic complexity som är 11, vilket innebär att man kan ta 11 olika vägar genom koden (dvs. control flow). McCabe rekommenderar att man inte ligger högre än 10. Alltså är vår kod lite för komplex. Det kan dock i vissa fall vara okej med en komplexitet upp till 15, men i vårt fall är det antagligen något som borde åtgärdas.

Class metrics

	Average operation complexity	Weighted method complexity
rougelikeclasses.WorldMap	2.83	17

På klassnivå ser det okej ut i denna version.

Statiska mått

Nya versionen

Method metrics

	essential cyclomatic complexity(G)	design complexity(G)	cyclomatic complexity(G)
rougelikeclasses.WorldMap.fillMap()	2	5	11

Den har en cyclomatic complexity som är 11, vilket innebär att man kan ta 11 olika vägar genom koden (dvs. control flow). McCabe rekommenderar att man inte ligger högre än 10. Alltså är vår kod lite för komplex. Det kan dock i vissa fall vara okej med en komplexitet upp till 15, men i vårt fall är det antagligen något som borde åtgärdas.

Class metrics

	Average operation complexity	Weighted method complexity
rougelikeclasses.WorldMap	4.43	31

Det här tyder på att det är för många metoder i denna klass och att den borde brytas ner i fler klasser. Det har vi också funderat på att göra innan vi gjorde testet, så att använda metrics kan hjälpa en att fatta mer välgrundade beslut.

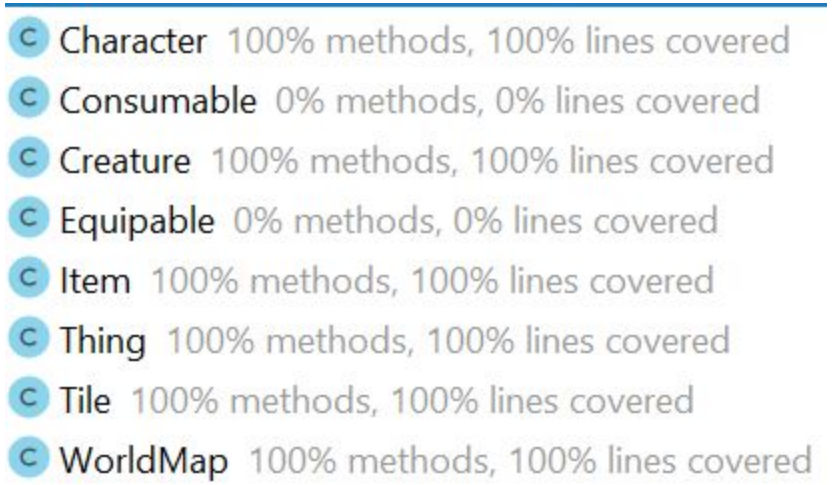
Statiska mått

Det bör påpekas att vi har fler rader källkod än testkod. Det bör vara tvärtom.

Detta beror på att vi från början inte använda oss tillräckligt mycket av testdesigntekniker när vi tog fram våra testfall.

Täckningsgrad

Nya:



Täckningsgrad

Gamla:

WorldMap
playerMoveBlocked är inte täckt, pga random tester.

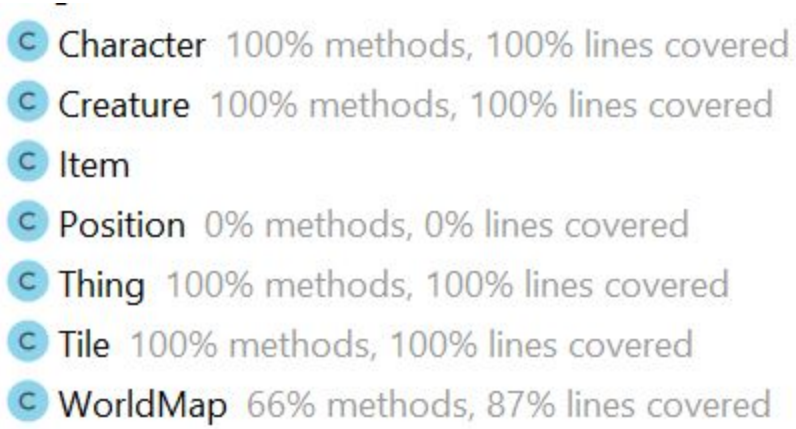
moveCreature är inte helt täckt, men de förändras pga random movement vid varje körning.

getHeight och getWidth används inte. Trodde att det skulle behövas.

Equipable
Finns en klass, men den är ej implementerad.

Consumable
Samma sak gäller som för Equipable.

Position
Ej implementerad.



Profiler

Vi gjorde ett test där vår player gick 1 miljon steg och slåss mot en creature. Vi tyckte att det var intressant att Profilern kunde visa oss att det gick åt enormt mycket minne till att skapa världskartan.

Detta hade kanske kunnat lösas genom att exempelvis byta datastruktur.

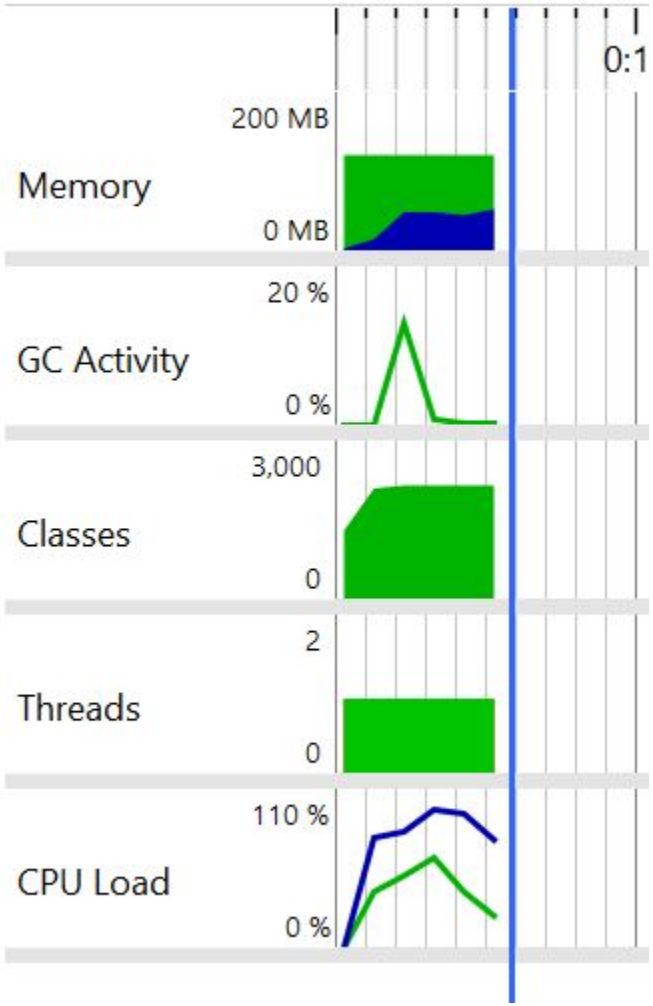
Det nedre diagrammet visar tidsförloppet. Vi kan se hur det gick åt mycket minne inledningsvis för att bygga och befolka kartan. Senare i tiden gick det åt ännu mer minne, för saker såsom System.out.prints när spelaren försöker gå in i blocked tiles. Detta kunde vara värt att åtgärda i en framtida version.

Profiler

Aggregation level: Classes

Name	Instance Count	Size
rougelikeclasses.Tile	1,000,001	32,000 kB
java.nio.HeapCharBuffer	75,544	3,626 kB
byte[ ]	64,748	2,893 kB
java.lang.String	62,333	1,495 kB
java.lang.Object[ ]	53,155	2,185 kB
int[ ]	51,830	6,353 kB
java.util.Formatter$FixedString	50,360	1,611 kB
java.util.Formatter$Flags	25,190	403 kB
java.util.Formatter$Flags[ ]	25,180	1,007 kB
java.util.Formatter$FormatSpecifier	25,180	1,007 kB
java.util.regex.IntHashSet[ ]	25,180	402 kB
java.util.regex.Matcher	25,180	1,812 kB
java.util.HashMap$Node	3,466	110 kB
java.util.concurrent.ConcurrentHashMap$Node	3,269	104 kB
java.lang.Class	2,579	308 kB
java.lang.Class[ ]	1,397	37,296 bytes
java.lang.invoke.MemberName	1,043	50,064 bytes
java.lang.Object	1,009	16,144 bytes
rougelikeclasses.Tile[ ]	1,000	4,016 kB
java.lang.invoke.ResolvedMethodName	873	20,952 bytes
java.lang.invoke.MethodType	864	34,560 bytes
java.lang.invoke.MethodType$ConcurrentWeakInternSet$W...	798	25,536 bytes
jdk.internal.org.objectweb.asm.SymbolTable$Entry	717	40,152 bytes
java.util.HashMap$Node[ ]	583	61,216 bytes
java.lang.StringBuilder	555	13,320 bytes
Total:	1,519,516	60,452 kB

Profiler


0:1
200 MB
Memory
0 MB
20 %
GC Activity
0 %
3,000
Classes
0
2
Threads
0
110 %
CPU Load
0 %

Byggscript

Vår POM-fil behövde inte ändras dels eftersom vi redan inledningsvis följde Mavens specifikation för filstruktur och för att det helt enkelt fortsatte fungera när den väl hade börjat fungera.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>se.su.dsv.InteProject</groupId>
    <artifactId>InteRouge</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <junit-jupiter-version>5.5.2</junit-jupiter-version>
        <maven.compiler.source>12</maven.compiler.source>
        <maven.compiler.target>12</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>



    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit-jupiter-version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>


</project>
```

Övrigt

När vi närmade oss slutet av projektet insåg vi att vi borde arbetat ännu mer TDD-baserat än vad vi gjorde ursprungligen. Det blev som det blev för att vi trodde att vi faktiskt jobbade TDD-baserat och gjorde rätt från början. Egentligen använde vi oss mer av felgissning än lämpliga testdesigntekniker.

Hade vi arbetat helt rätt från början så hade vi kanske hunnit komma längre i arbetet och kunnat implementerat fler metoder.

Övrigt

En sak som kan poängteras är velocity-variablen i Creature. Den finns för att kravspec:en bad som en sådan men vi har inte implementerat variablen utan istället implementerat speed som en funktion av rörelse.

En annan är att vi insåg att Move-metoderna borde inte ligga i worldMap utan i en egen klass, vilket också indikerades av McCabe-måtten.

Vi har dessutom väldigt få tester för creatureMovement och testar därför inte alla möjligheter. Vi borde ha tagit ut fler från beslutstabellen. Exempelvis har vi ingen som testar när en creature bara rör på sig eller om man skickar in 2 för rörelse.